INF 4300
21.10.15

Multivariate classification

Anne Solberg (anne@ifi.uio.no)

Based on Chapter 2 (2.1-2.6) in Duda and Hart: Pattern Classification

Bayes rule for a classification problem

- Suppose we have J, j=1,...J classes. ω is the class label for a pixel, and $\boldsymbol{x}$ is the observed feature vector).
- We can use Bayes rule to find an expression for the class with the highest probability:

$$P(\omega_j \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid \omega_j) P(\omega_j)}{p(\mathbf{x})}$$

$$\text{posterior probability} = \frac{\textit{likelihood} \times \text{prior probability}}{\text{normalizing factor}}$$

- $P(\omega_j)$ is the prior probability for class ω_j. If we don't have special knowledge that one of the classes occur more frequent than other classes, we set them equal for all classes. ($P(\omega_j)=1/J$, j=1.,,,J).

Bayes rule explained

$$P(\omega_j \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid \omega_j) P(\omega_j)}{p(\mathbf{x})}$$

- $p(\mathbf{x}|\omega_j)$ is the probability density function that models the likelihood for observing feature vector **x** if the pixel belongs to class $\omega_{j.}$
  - Typically we assume a type of distribution, e.g. Gaussian, and the mean and covariance of that distribution is fitted to some data that we know belong to that class. This fitting is called classifier training.
- $P(\omega_j|\mathbf{x})$ is the posterior probability that the pixel actually belongs to class ω_j given the observed feature vector **x**.
- $p(\mathbf{x})$ is just a scaling factor that assures that the probabilities sum to 1.

The conditional density $p(\mathbf{x}|\ \omega_s)$

- Any probability density function can be used to model $p(\mathbf{x}|\ \omega_s)$
- A common model is the multivariate Gaussian density.
- The multivariate Gaussian density:

$$p(\mathbf{x} \mid \omega_s) = \frac{1}{(2\pi)^{n/2} \left|\boldsymbol{\Sigma}_s\right|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_s)^t \boldsymbol{\Sigma}_s^{-1} (\mathbf{x} - \boldsymbol{\mu}_s)\right]$$

- If we have d features, μ_s is a vector of length d and and Σ_s a d×d matrix (depends on class *s*)

$$\boldsymbol{\mu}_S = \begin{bmatrix} \mu_{1s} \\ \mu_{2s} \\ \\ \\ \mu_{ns} \end{bmatrix} \qquad \boldsymbol{\Sigma}_S = \begin{bmatrix} \sigma_{11} & \sigma_{12} & . & . & \sigma_{1n} \\ \sigma_{21} & \sigma_{22} & . & . & . \\ \sigma_{31} & \sigma_{11} & . & . & . \\ . & . & . & . & . \\ \sigma_{n1} & \sigma_{n2} & . & \sigma_{nn-1} & \sigma_{nn} \end{bmatrix}$$

Symmetric d×d matrix
σ_{ii} is the variance of feature i
σ_{ij} is the covariance between feature i and feature j
Symmetric because $\sigma_{ij} = \sigma_{ji}$

- $|\Sigma_s|$ is the determinant of the matrix $\Sigma_{s,}$ and Σ_s^{-1} is the inverse

Mean vectors and covariance matrices in d dimensions

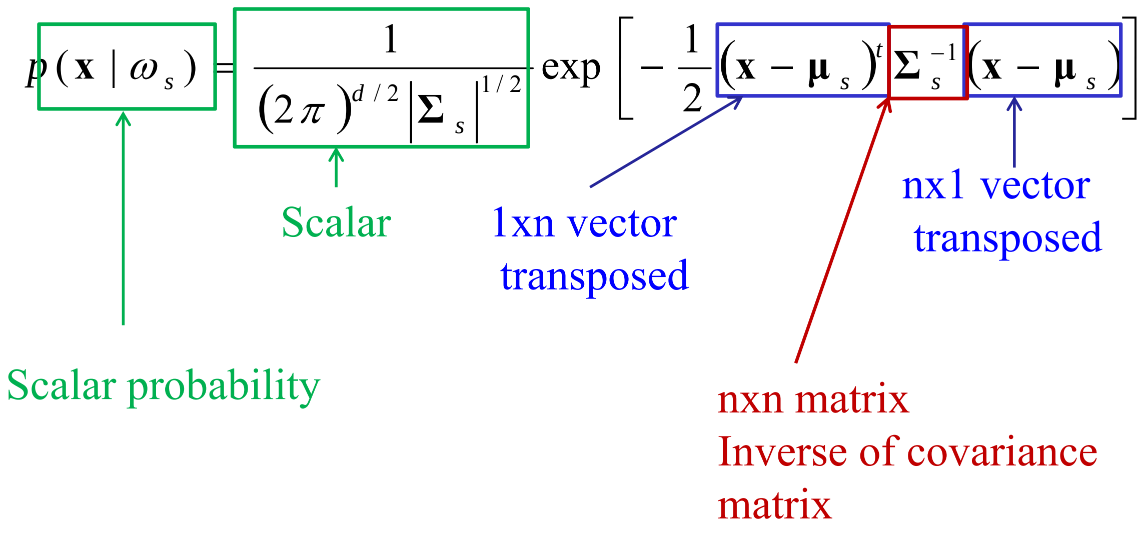
- If **x** is a d-dimensional feature vector for one object/pixel, we can formulate its mean vector and covariance matrix as:

$$\mathbf{x}=\begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ x_d \end{bmatrix} \qquad \boldsymbol{\mu}=E[\mathbf{x}]=\begin{bmatrix} E(x_1) \\ E(x_2) \\ . \\ . \\ E(x_d) \end{bmatrix}=\begin{bmatrix} \mu_1 \\ \mu_2 \\ . \\ . \\ \mu_d \end{bmatrix}$$

$$\boldsymbol{\Sigma}=\begin{bmatrix} \sigma_{11} & \sigma_{12} & . & . & \sigma_{1d} \\ \sigma_{21} & \sigma_{22} & . & . & \sigma_{d} \\ . & . & . & . & . \\ . & . & . & . & . \\ \sigma_{d1} & \sigma_{d2} & . & . & \sigma_{dd} \end{bmatrix}=\begin{bmatrix} \sigma_1^2 & \sigma_{12} & . & . & \sigma_{1d} \\ \sigma_{21} & \sigma_2^2 & . & . & \sigma_{2d} \\ . & . & . & . & . \\ . & . & . & . & . \\ \sigma_{d1} & \sigma_{d2} & . & . & \sigma_d^2 \end{bmatrix}$$

- with d features, the mean vector μ will be of size 1xd and Σ of size dxd.

Inspecting p(**x**|ωs)



$$\boxed{p(\mathbf{x}\mid\omega_s)}=\boxed{\frac{1}{(2\pi)^{d/2}\left|\boldsymbol{\Sigma}_s\right|^{1/2}}}\exp\left[-\frac{1}{2}\boxed{(\mathbf{x}-\boldsymbol{\mu}_s)^t}\boxed{\boldsymbol{\Sigma}_s^{-1}}\boxed{(\mathbf{x}-\boldsymbol{\mu}_s)}\right]$$

Scalar

1xn vector transposed

nx1 vector transposed

Scalar probability

nxn matrix
Inverse of covariance matrix

The mean vectors μ_s for each class

- The mean vector for class s is defined as the expected value of **x**:

$$\boldsymbol{\mu}_s = E[\mathbf{x}] = \begin{bmatrix} E(x_1) \\ E(x_2) \\ . \\ . \\ E(x_d) \end{bmatrix} = \begin{bmatrix} \mu_1{}^s \\ \mu_2{}^s \\ . \\ . \\ \mu_d{}^s \end{bmatrix}$$

← class s

← feature number d

- with d features, the mean vector μ will be of size 1xd.
- If we have M_s training samples that we know belong to class s, we can estimate the mean vector as:

$$\hat{\boldsymbol{\mu}}_s = \frac{1}{M_s} \sum_{m=1}^{M_s} \mathbf{x}_m ,$$

where the sum is over all training samples belonging to class s

The covariance matrix Σ_s for each class

- The covariance for class s is defined as the expected value of (**x**-μ)(**x**-μ)^t:

$$\boldsymbol{\Sigma}_s = \begin{bmatrix} \sigma_{11} & \sigma_{12} & . & . & \sigma_{1d} \\ \sigma_{21} & \sigma_{22} & . & . & \sigma_{d} \\ . & . & . & . & . \\ . & . & . & . & . \\ \sigma_{d1} & \sigma_{d2} & . & . & \sigma_{dd} \end{bmatrix} = \begin{bmatrix} \sigma_1^2 & \sigma_{12} & . & . & \sigma_{1d} \\ \sigma_{21} & \sigma_2^2 & . & . & \sigma_{2d} \\ . & . & . & . & . \\ . & . & . & . & . \\ \sigma_{d1} & \sigma_{d2} & . & . & \sigma_d^2 \end{bmatrix}$$

- with d features, the covariance matrix Σ_s will be of size dxd.
- If we have M_s training samples that we know belong to class s, we can estimate the covariance matrix Σ_s . (The estimate of a random variable f is denoted $\hat{f}$)

$$\hat{\Sigma}_s = \frac{1}{M_s} \sum_{m=1}^{M_s} (\mathbf{x}_m - \hat{\boldsymbol{\mu}}_s)(\mathbf{x}_m - \hat{\boldsymbol{\mu}}_s)^t$$

where the sum is over all training samples belonging to class s

- Each term σ_{ij} is computed as:

$$\sigma_{ij,s}{}^2 = \frac{1}{M_s} \sum_{m=1}^{M_s} (x_{m,i} - \hat{\mu}_{i,s})(x_{m,j} - \hat{\mu}_{j.s})^t$$

for the covariance between feature i and j for class s

More on the covariance matrix Σ_s

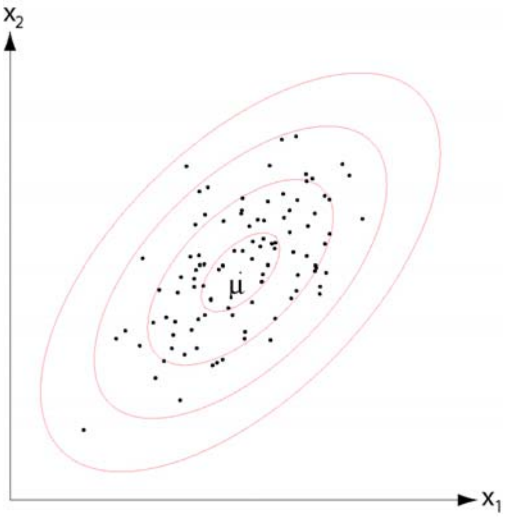
- The covariance matrix Σ_s will always be symmetric and positive semidefinite.
- If all components of x have non-zero variance, Σ_s will be positive definite.
- σ_{ij} is the covariance between features *i* and *j*.
- If features x_i and x_j are uncorrelated, $\sigma_{ij} = 0$.
- In the general case, Σ_s will have d(d+1)/2 different values.

A 2D Gaussian model

- Parameters $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ define a density as a ”bump”
- The curves on the plot are contours of equal probability, just as the contours on a map
- The matrix $\boldsymbol{\Sigma}$ in this case has three different elements, variance in each of the axes, and covariance between the axes

$$\Sigma_S = \begin{bmatrix} \sigma_{11}^2 & \sigma_{12}^2 \\ \sigma_{21}^2 & \sigma_{22}^2 \end{bmatrix}$$

- σ_{11}^2 is the variance for feature 1
- $\sigma_{12}=\sigma_{21}$ is the covariance between feature 1 and 2
- σ_{22}^2 is the variance for feature 2



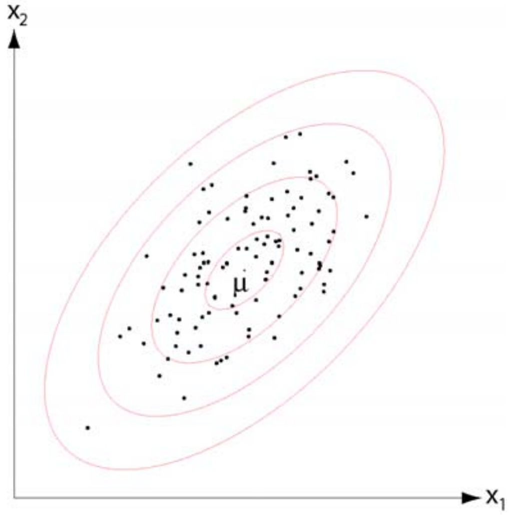
The covariance matrix and ellipses

- In 2D, the Gaussian model can be thought of as approximating the classes in 2D feature space with ellipses.
- The mean vector $\mu=[\mu_1, \mu_2]$ defines the the center point of the ellipses.
- σ_{12}, the covariance between the features defines the orientation of the ellipse.
- σ_{11} and σ_{22} defines the width of the ellipse.

$$\Sigma_S = \begin{bmatrix} \sigma_{11} & \sigma_{12} \\ \sigma_{21} & \sigma_{22} \end{bmatrix}$$

- The ellipse defines points where the probability density is equal
  - Equal in the sense that the distance to the mean as computed by the Mahalanobis distance is equal.
  - The Mahalanobis distance between a point x and the class center μ is:

$$r^2 = (x-\mu)^T \Sigma^{-1} (x-\mu)$$



The main axes of the ellipse is determined by the eigenvectors of Σ.
The eigenvalues of Σ gives their length.

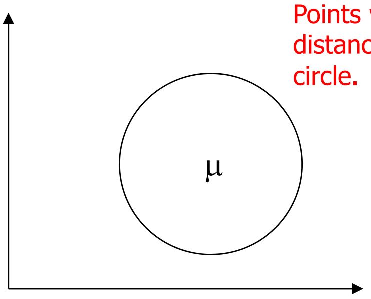
Euclidean distance vs. Mahalanobis distance

- Euclidean distance between point x and class center μ:

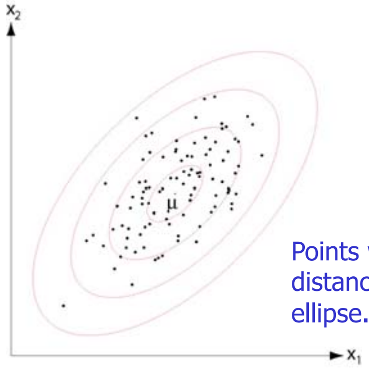
$$(x-\mu)^T (x-\mu) = \|x-\mu\|^2$$

- Mahalanobis distance between x and μ:

$$r^2 = (x-\mu)^T \Sigma^{-1} (x-\mu)$$



Points with equal distance to μ lie on a circle.



Points with equal distance to μ lie on an ellipse.

Discriminant functions for the normal density

- We saw last lecture that the minimum-error-rate classification can be computed using the discriminant functions

$$g_i(\mathbf{x}) = \ln p(\mathbf{x} \mid \omega_i) + \ln P(\omega_i)$$

- With a multivariate Gaussian we get:

$$g_i(\mathbf{x}) = -\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)^t \Sigma_i^{-1}(\mathbf{x}-\boldsymbol{\mu}_i) - \frac{d}{2}\ln 2\pi - \frac{1}{2}\ln|\Sigma_i| + \ln P(\omega_i)$$

- Let ut look at this expression for some special cases:

Case 1: $\Sigma_j=\sigma^2 I$

- In this case we assume that the features are uncorrelated (independent) with the same variance σ^2
- The covariances $\sigma_{ij}=0$ (by definition if the features are uncorrelated).
- The discriminant functions can be expressed as:

$$g_i(\mathbf{x}) = -\frac{\|\mathbf{x}-\boldsymbol{\mu}_i\|^2}{2\sigma^2} + \ln P(\omega_i)$$
$$\text{where } \|\mathbf{x}-\boldsymbol{\mu}_i\|^2 = (\mathbf{x}-\boldsymbol{\mu}_i)^t(\mathbf{x}-\boldsymbol{\mu}_i)$$

- Thus we model the probabilities as n-dimensional *spheres* because points that have equal discriminant function will lie on a circle around the mean μ_i.
- $\boldsymbol{\Sigma}_j^{-1}=\mathbf{I}/\sigma^2$
- $|\boldsymbol{\Sigma}_j|= \sigma^{2n}$

Case 1: $\Sigma_j=\sigma^2 I$

- The discriminant functions simplifies to **linear** functions using such a shape on the probability distributions

$$
\begin{aligned}
g_j(\mathbf{x}) &= -\frac{1}{2(\sigma^2 I)}(\mathbf{x}-\boldsymbol{\mu}_j)^T(\mathbf{x}-\boldsymbol{\mu}_j) - \frac{d}{2}\ln(2\pi) - \frac{1}{2}\ln\left|\sigma^2 I\right| + \ln P(\omega_j) \\
&= -\frac{1}{2(\sigma^2 I)}(\mathbf{x}^T\mathbf{x} - 2\boldsymbol{\mu}_j^T\mathbf{x} + \boldsymbol{\mu}_j^T\boldsymbol{\mu}_j) - \frac{d}{2}\ln(2\pi) - \frac{1}{2}\ln\left|\sigma^2 I\right| + \ln P(\omega_j)
\end{aligned}
$$

Common for all classes, no need to compute these terms
Since ***x^T^x*** is common for all classes, an equivalent $g_j(\mathbf{x})$ is a linear function of ***x:*** .

$$
\frac{1}{(\sigma^2)}\boldsymbol{\mu}_j^T\mathbf{x} - \frac{1}{2(\sigma^2)}\boldsymbol{\mu}_j^T\boldsymbol{\mu}_j + \ln P(\omega_j)
$$

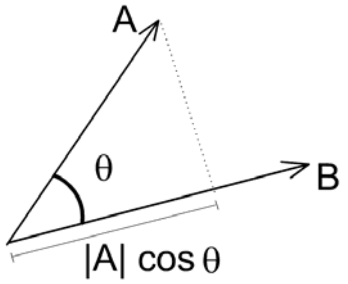
Linear algebra basics: Inner product between two vectors.

- The inner product (or dot product) between two vectors (of length N)a and b or is given by

$$
\langle a,b\rangle = \sum_{i=1}^{N} a_i b_i = a^T b
$$

- The angle between two vectors A and B is defined as:

$$
\cos\theta = \frac{\langle A,B\rangle}{\|A\|\|B\|}
$$



- If the inner product of two vectors is zero, they are normal to each other.

Case 1: $\Sigma_j=\sigma^2 I$

- Now we get an equivalent formulation of the discriminant functions:

$$g_i(\mathbf{x}) = \mathbf{w}_i^t\mathbf{x} + w_i 0$$

$$\text{where } \mathbf{w}_i = \frac{1}{\sigma^2}\boldsymbol{\mu}_i \text{ and } wi0 = -\frac{1}{2\sigma^2}\boldsymbol{\mu}_i^t\boldsymbol{\mu}_i + \ln P(\omega_i)$$

- An equation for the decision boundary $g_i(\mathbf{x})=g_j(\mathbf{x})$ can be written as

$$\mathbf{w}^t(\mathbf{x}-\mathbf{x}_0) = 0$$

$$\text{where } \mathbf{w} = \boldsymbol{\mu}_\text{i} - \boldsymbol{\mu}_\text{j}$$

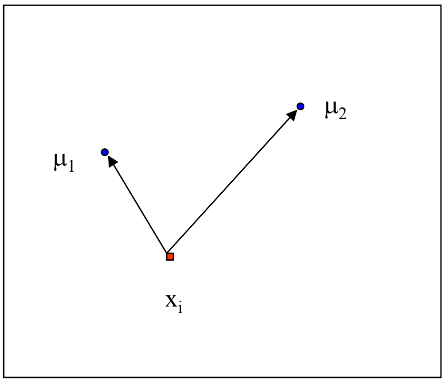
$$\text{and } x_0 = \frac{1}{2}\left(\boldsymbol{\mu}_\text{i} - \boldsymbol{\mu}_\text{j}\right) - \frac{\sigma^2}{\left\|\boldsymbol{\mu}_\text{i} - \boldsymbol{\mu}_\text{j}\right\|^2}\ln\frac{P(\omega_i)}{P(\omega_j)}\left(\boldsymbol{\mu}_\text{i} - \boldsymbol{\mu}_\text{j}\right)$$

- $\mathbf{w}=\mu_i-\mu_j$ is the vector between the mean values.
- This equation defines a hyperplane through the point x_0, and orthogonal to **w**.
- If $P(\omega_i)=P(\omega_j)$ the hyperplane will be located halfway between the mean values.
- Proving this involves some algebra, see the proof at https://www.byclb.com/TR/Tutorials/neural_networks/ch4_1.htm

- If the features were indepenent ($\Sigma_j=\sigma^2 I$) the discriminant function was simplified to:

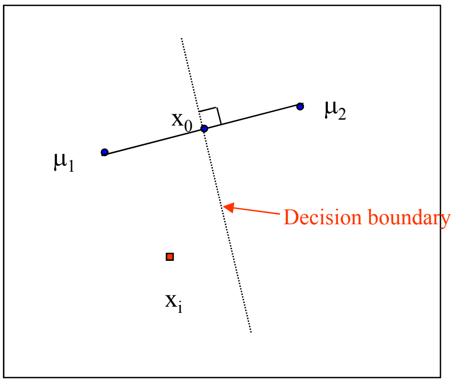
$$g_j^{'}(\mathbf{x}) = -\frac{1}{2\sigma^2}(\mathbf{x}-\boldsymbol{\mu}_j)^T(\mathbf{x}-\boldsymbol{\mu}_j) + \ln P(\omega_j)$$

$$= -\frac{1}{2\sigma^2}\left\|\mathbf{x}-\boldsymbol{\mu}_j\right\|^2 + \ln P(\omega_j)$$

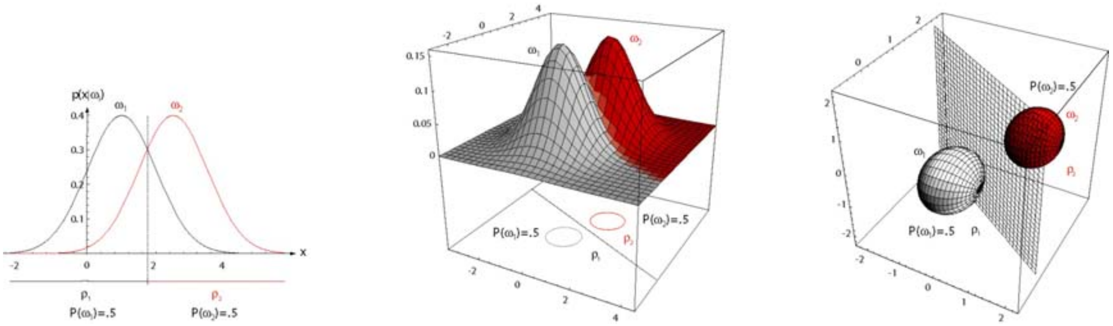


- This results in linear decision boundaries.
- Computing this discriminant function to classify pattern x_i involves computing the distance from the point to the mean values μ_s for each class.

- The discriminant function (when $\Sigma_j=\sigma^2 I$) that defines the border between class 1 and 2 in the feature space is a straight line.
- The discriminant function intersects the line connecting the two class means at the point $x_0=(\mu_1- \mu_2)/2$ (if we do not consider prior probabilities).
- The discriminant function will also be normal to the line connecting the means.



A simple model, $\Sigma_j=\sigma^2 I$



- The distributions are spherical in *d* dimensions.
- The decision boundary is a generalized hyperplane of *d-1* dimensions
- The decision boundary is perpendicular to the line separating the two mean values
- This kind of a classifier is called a linear classifier, or a linear discriminant function
  - Because the decision function is a linear function of ***x***.
- If $P(\omega_i)= P(\omega_j)$, the decision boundary will be half-way between μ_i and μ_j

Minimum distance classification

- If all classes have equal prior probabilities, x_0 will be the point halfway between the mean vectors.
- Classification will consist of assigning feature vector x to the same class as the closest mean measured by Euclidean distance $||x-\mu_i||$.
- A classifier based on the Euclidean distance is called a **minimum distance classifier**.

Case 2: Common covariance, $\Sigma_j = \Sigma$

- If we assume that all classes have the same shape of data clusters, an intuitive model is to assume that their probability distributions have the same shape
- By this assumption we can use all the data to estimate the covariance matrix
- This estimate is common for all classes, and this means that also in this case the discriminant functions become linear functions

$$g_j(\mathbf{x}) = -\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_j)^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_j) - \frac{1}{2}\ln|\boldsymbol{\Sigma}| + \ln P(\omega_j)$$

$$= -\frac{1}{2(\sigma^2 I)}(\mathbf{x}^T \boldsymbol{\Sigma}^{-1}\mathbf{x} - 2\boldsymbol{\mu}_j^T \boldsymbol{\Sigma}^{-1}\mathbf{x} + \boldsymbol{\mu}_j^T \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_j) - \frac{1}{2}\ln|\boldsymbol{\Sigma}| + \ln P(\omega_j)$$

Common for all classes, no need to compute

Since $\boldsymbol{x}^T\boldsymbol{x}$ is common for all classes, $g_j(\boldsymbol{x})$ again reduces to a linear function of $\boldsymbol{x}$.

Case 2: Common covariance, $\Sigma_j= \Sigma$

- An equivalent formulation of the discriminant functions is

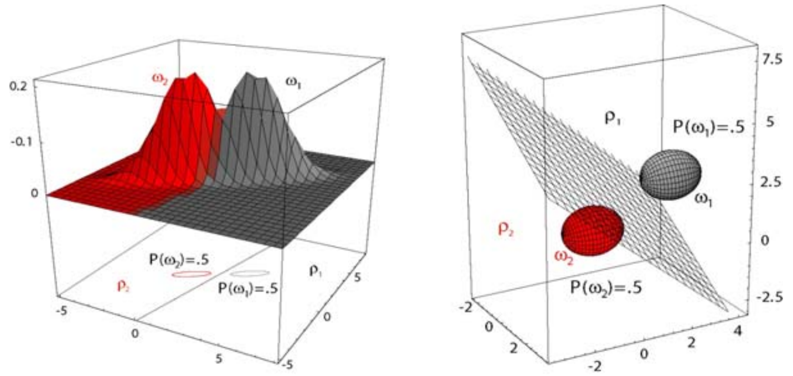
$$
\begin{aligned}
&g_i(\mathbf{x}) = \mathbf{w}_i^t\mathbf{x} + wi_0 \\
&\text{where } \mathbf{w}_i = \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_i \\
&\text{and } wi_0 = -\frac{1}{2}\boldsymbol{\mu}_i{}^t\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_i + \ln P(\omega_i)
\end{aligned}
$$

- The decision boundaries are again hyperplanes.
- The decision boundary has the equation:

$$
\begin{aligned}
&\mathbf{w}^T(\mathbf{x}-\mathbf{x}_0) = 0 \\
&\mathbf{w} = \Sigma^{-1}(\mu_i - \mu_j) \\
&x_0 = \frac{1}{2}(\mu_i + \mu_j) - \frac{\ln\left[(P(\omega_i)/(P(\omega_j)\right]}{(\mu_i-\mu_j)^T\Sigma^{-1}(\mu_i-\mu_j)}(\mu_i-\mu_j)
\end{aligned}
$$

- Because $\mathbf{w}_i$= $\boldsymbol{\Sigma}^{-1}(\mu_i- \mu_j)$ is not in the direction of $(\mu_i- \mu_j)$, the hyperplane will not be orthogonal to the line between the means.

Common covariance, $\Sigma_j= \Sigma$



- The classes can be described by hyperellipsoides in *d* dimensions.
- All hyperellipsoids have the same orientation.
- The decision boundary will again be a hyperplane.
- Because ***w=*** $\Sigma^{-1}(\mu_i$-$\mu_j)$ is generally not in the direction of μ_i-μ_j, the hyperplane will not be perpendicular to the line between the means.
- Consider a point x_0 on the line μ_i-μ_j, defined by the prior probabilities:
  - If $P(\omega_i)= P(\omega_j)$, x_0 will be half way between the means.
  - The separating hyperplane will *intersect* the line at x_0

Case 3:, Σ_j=arbitrary

- When all classes are modeled as having different *shapes,* the discriminant functions cannot be simplified
- This means that the discriminant functions will be *quadratic* functions
- Decision boundaries will be hyperquadrics and assume any of the general forms:
  - hyperplanes, pairs of hyperplanes, hyperspheres, hyperellisoides, hyperparaboloids, hyperhyperboloids...

Case 3:, Σ_j=arbitrary

- The discriminant functions will be quadratic:

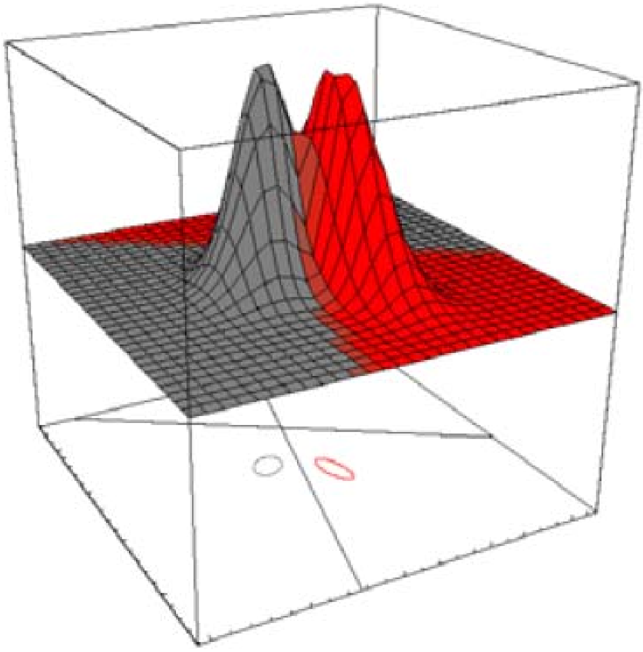
$$g_i(\mathbf{x}) = \mathbf{x}^t \mathbf{W}_i \mathbf{x} + \mathbf{w}_i^t \mathbf{x} + wi_0$$

$$\text{where } \mathbf{W}_i = -\frac{1}{2}\mathbf{\Sigma}_i^{-1}, \qquad \mathbf{w}_i = \mathbf{\Sigma}_i^{-1}\boldsymbol{\mu}_i$$

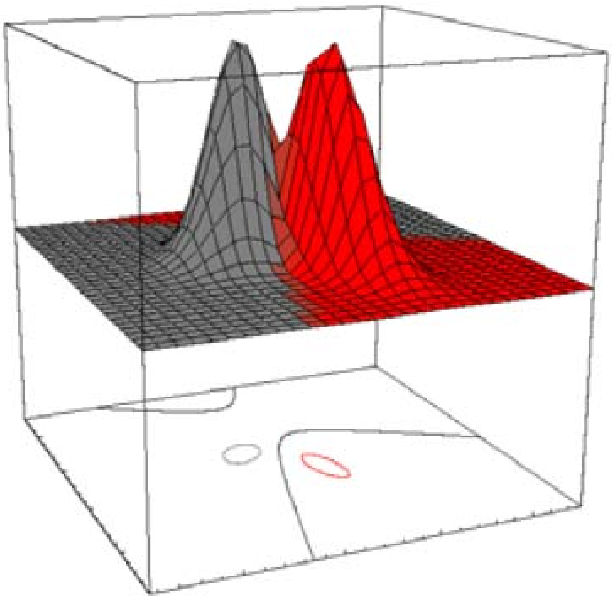
$$\text{and } wi_0 = -\frac{1}{2}\boldsymbol{\mu}_i{}^t \mathbf{\Sigma}_i^{-1}\boldsymbol{\mu}_i - \frac{1}{2}\ln\left|\mathbf{\Sigma}_i\right| + \ln P(\omega_i)$$

- The decision surfaces are hyperquadrics and can assume any of the general forms:
  - hyperplanes
  - hypershperes
  - pairs of hyperplanes
  - hyperellisoids,
  - Hyperparaboloids,..
- The next slides show examples of this.
- In this general case we cannot intuitively draw the decision boundaries just by looking at the mean and covariance.

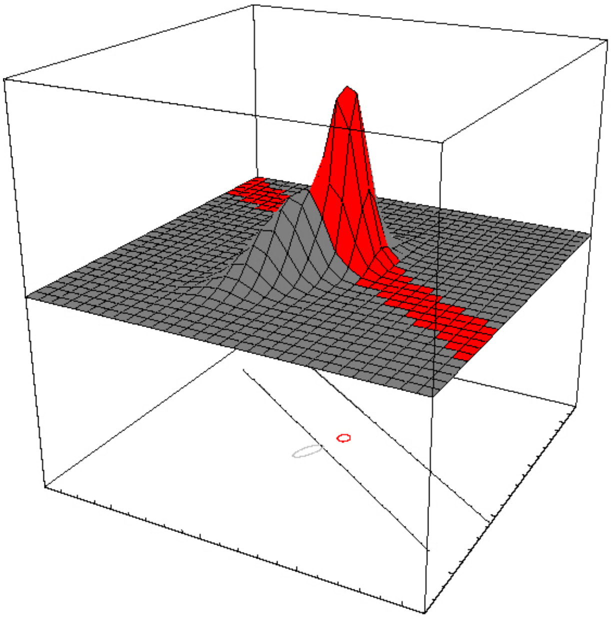
The full model, Σ_j=arbitrary - example



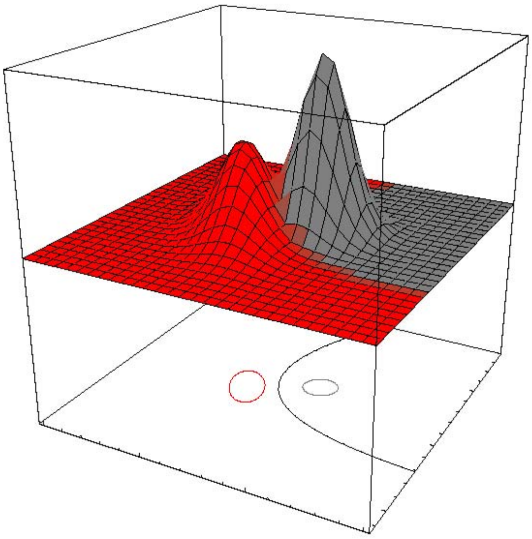
The full model, Σj=arbitrary - example



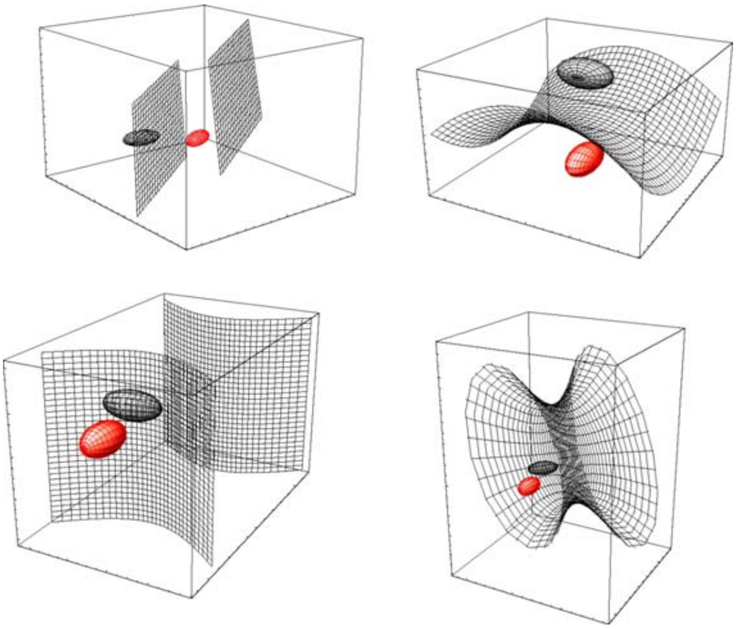
The full model, Σ_j=arbitrary - example



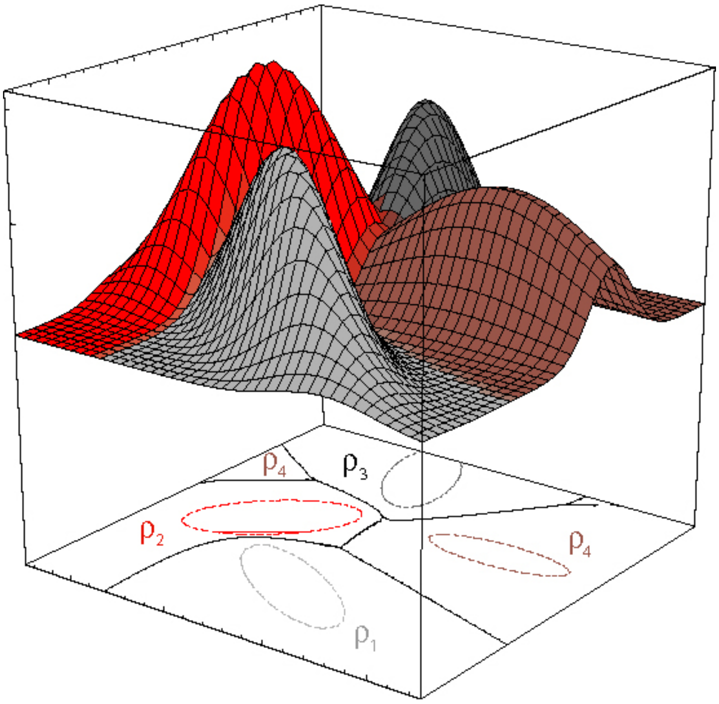
The full model, Σj=arbitrary - example



The full model, Σj=arbitrary - example



A multiclass example



Is the Gaussian classifier the only choice?

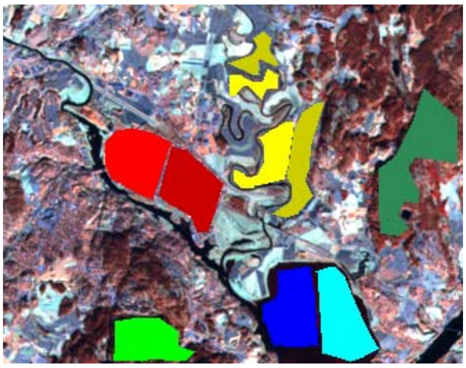
- The Gaussian classifier gives linear or quadratic discriminant function.
- Other classifiers can give arbitrary complex decision surfaces (often piecewise-linear)
  - Mixtures of Gaussians
  - Other probability density functions (t-distribution, exponetial distributions).
  - Neural networks
  - Support vector machines (INF 5300)
  - Ensembles of simple classifiers

    ADAboost

    Random forest/decision trees
  - kNN (k-Nearest-Neighbor) classification (next week)

Using masks to train and test

- Training mask: a mask where regions to train each class are marked using different pixel values, e.g. class label=1 for class 1, 2 for class 2 etc.
- Test mask: a similar mask as training, but to estimate classifier accuracy only.



Training a classifier

- Obtain as many ground truth samples for each class as possible
  - If visual inspection is reliable, experts can mark training regions interactively.
  - For remote sensing, go out in the field and collect field samples (or use images from a different sensor)
  - For symbol recognition, mark a set of symbols manually.
  - For medical applications, use e.g. tissue samples or interpretations made by experts.
- Divide the ground truth into a training set and a test set.
- Use feature extraction and feature selection/evaluation to determine the best set of features.

- Decide if a linear or quadratic classifier is needed.

  $\hat{\mu}_s$ has n elements

  $\hat{\Sigma}_s$ has n(n-1)/2 elements
- For each class, compute μ_s (and Σ_s)using the given Maximum Likelihood estimates.

Classifying new data

- For each sample, compute the posterior probabilities for each class.

$$P(\omega_s \mid x) \propto p(x \mid \omega_s) P(\omega_s)$$

$$= \left[\frac{1}{(2\pi)^{P/2} |\Sigma_s|^{1/2}} \exp\left[-\frac{1}{2} (x - \mu_s)^t \Sigma_s^{-1} (x - \mu_s) \right] \right] P(\omega_s)$$

- Classify the sample to the class with the highest posterior probability.
- Evaluate the performance of the classifier.
- We can also produce images of the posterior probability for each class.

Validating classifier performance

- Classification performance is evaluated on a different set of samples with known class - the test set.
- The training set and the test set must be independent!
- Normally, the set of ground truth pixels (with known class) is partionioned into a set of training pixels and a set of test pixels of approximately the same size.
- This can be repeated several times to compute more robust estimates as average test accuracy over several different partitions of test set and training set.
  - By selecting e.g. 10 random partitions of the set of samples into a training set and a test set.

Confusion matrices

- A matrix with the true class label versus the estimated class labels for each class

Estimated class labels

True class labels

	Class 1	Class 2	Class 3	Total #samples
Class 1	80	15	5	100
Class 2	5	140	5	150
Class 3	25	50	125	200
Total	110	205	135	450

Confusion matrix - cont.

Alternatives:

- Report nof. correctly classified pixels for each class.
- Report the percentage of correctly classified pixels for each class.
- Report the percentage of correctly classified pixels in total.
  - Why is this not a good measure if the number of test pixels from each class varies between classes?

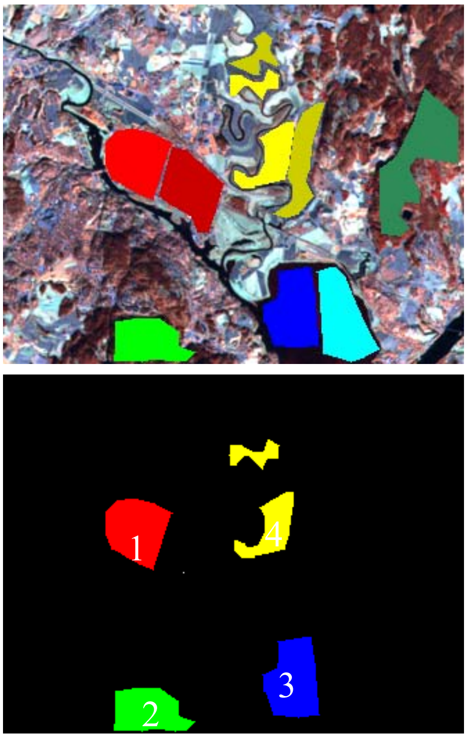
	Class 1	Class 2	Class 3	Total #samples
Class 1	80	15	5	100
Class 2	5	140	5	150
Class 3	25	50	125	200
Total	110	205	135	450

A classification example

Landsat image with 6 spectral bands
The 6 bands will be the features
Training areas and test areas shown in mask

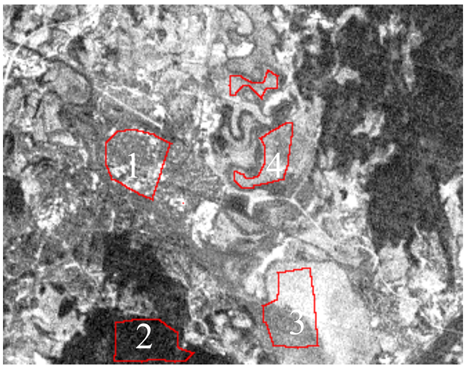
Upper part: RGB-false color image created from bands 4,5 and 6 with training and test regions overlaid.

Lower part: image of training regions only



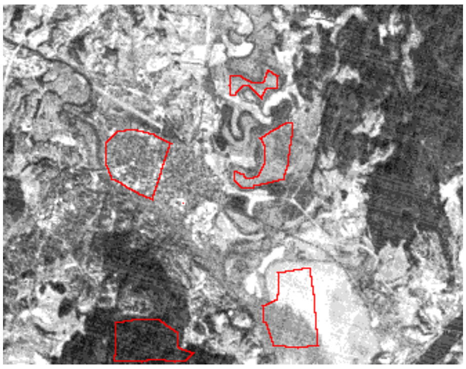
Visual inspection of feature 1

Class 2 (forest) seems to be well separated,
Maybe also class 1 (urban)



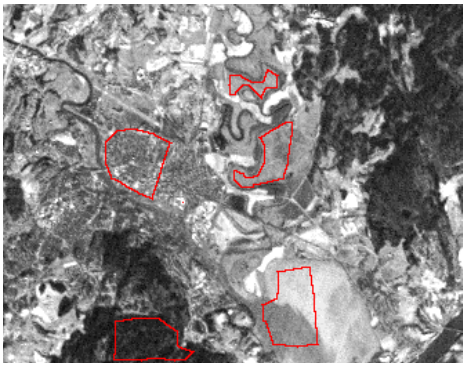
Visual inspection of feature 2

Class 2 (forest) seems to be well separated



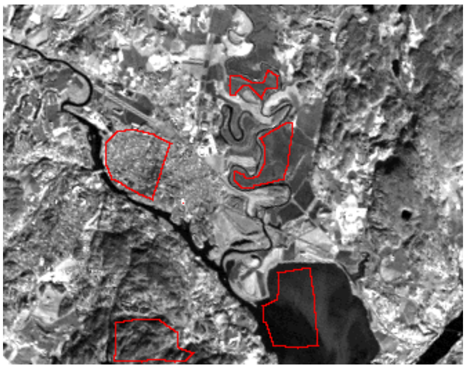
Visual inspection of feature 3

Class 2 (forest) seems to be well separated,
Class 1 (urban) seems to be well separated



Visual inspection of feature 4

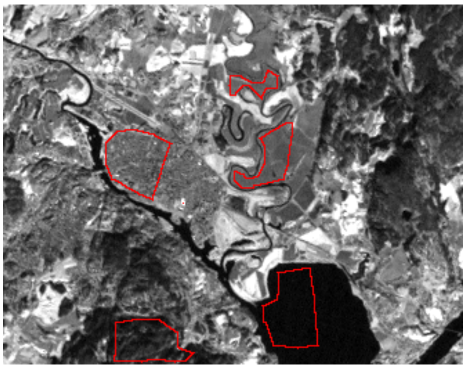
Class 1 (water) seems to be well separated,
Maybe also class 4 (agricultural)



Visual inspection of feature 5

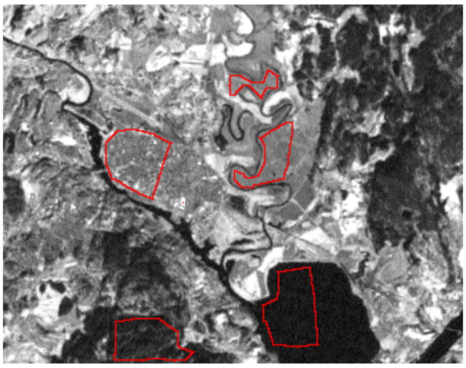
Water and forest appears similar - but the variance might be different

Urban and agricultural appears similar – but the variance might be different

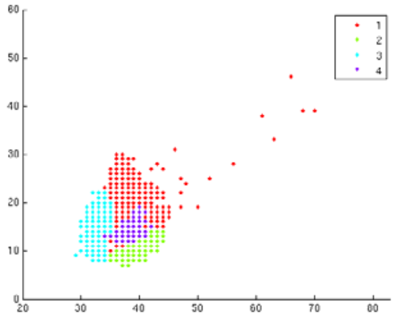


Visual inspection of feature 6

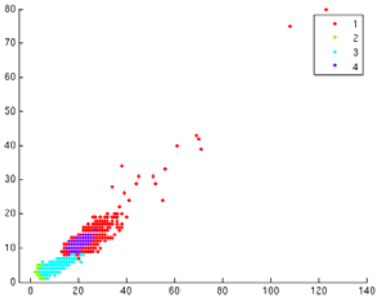
Seems similar to feature 5, but with better contrast



Selected scatter plots (gscatter)

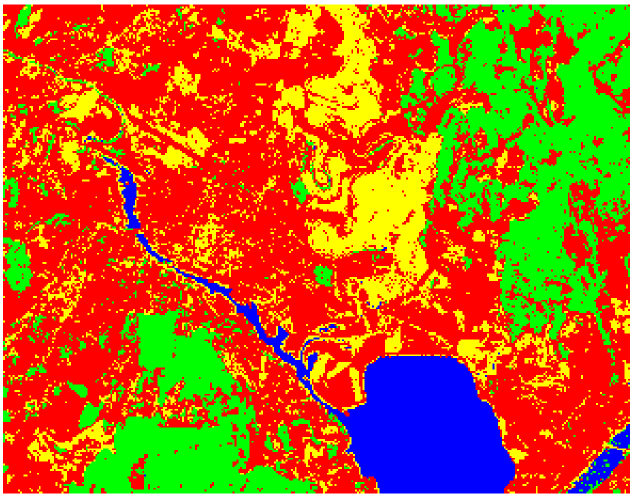


Scatterplot between feature 1 and 4



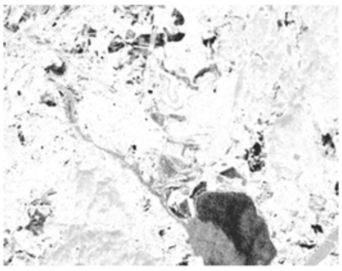
Scatterplot between feature 5 and 6

Classified images

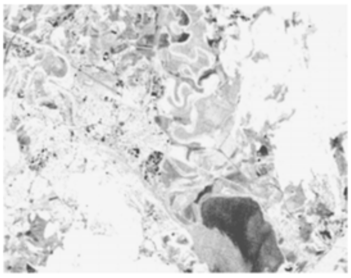


The entire image classified to the most probable class

Display the posterior probabilities as images



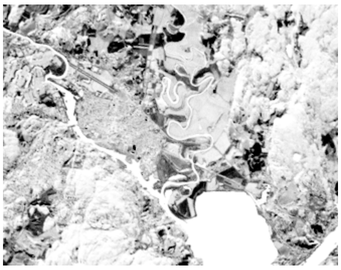
Posterior probability for class urban



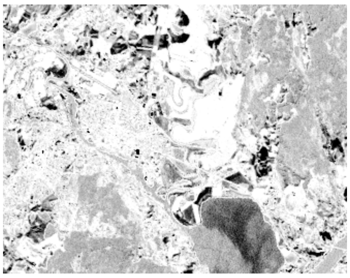
Posterior probability for class forest

Dark values:
Probabilities close to 0

Bright values:
Probabilities close to 1



Posterior probability for class water



Posterior probability for class agricultural

Confusion matrix for the training set

True class	Assigned to Class1	Assigned to Class2	Assigned to Class 3	Assigned to Class4
Class 1	1340	2	0	310
Class 1	43	1253	0	2
Class 3	0	0	1738	0
Class 4	131	3	0	1266

Accuracy per class:
Class1: 81%
Class2: 96%
Class3: 100%
Class4: 90%

Averaged over all classes: 91.7%

Confusion matrix for the test set

True class	Assigned to Class1	Assigned to Class2	Assigned to Class 3	Assigned to Class4
Class 1	1474	3	1	251
Class 1	513	2311	0	0
Class 3	14	0	1953	0
Class 4	213	2	0	1390

Accuracy per class:
Class1: 85%
Class2: 81%
Class3: 98%
Class4: 86%

Averaged over all classes: 87.5%

Learning goals from this lecture

- Be able to use and implement Bayes rule with a d-dimensional Gaussian distribution.
- Know how μ_s and Σ_s are estimated.
- Understand the 2-dimensional case where a covariance matrix is illustrated as an ellipse.
- Be able to simplify the general discriminant function for 3 cases.
- Have a geometric interpretation of classification with 2 features.